# Porting and tuning R5F code on Leonardo

**M. YAGI**

**National Institutes for Quantum Science and Technology (QST)**
**Rokkasho Institute for Fusion Energy**

# 5-field Reduced Drift MHD Model

**Model consists of Vorticity equation, Generalized Ohm's law, Continuity equation, Parallel momentum equation, Electron temperature evolution equation: $(\boldsymbol{F}, \boldsymbol{A}, \boldsymbol{n}, \boldsymbol{v}, \boldsymbol{T_e})$**

**It is an extension of 4-field reduced drift MHD model by Hazeltine and Meiss, Phys. Rep. 121 (1985)1-164.**

$$\frac{d}{dt}\nabla_\perp^2 F - \delta[\nabla_\perp p_i; \nabla_\perp F] = -\nabla_\parallel \nabla_\perp^2 A - [\Omega, p] + \mu_\perp^i \nabla_\perp^4 F$$

$$\frac{d}{dt}\left(A - \delta^2 \frac{m_e}{m_i}\nabla_\perp^2 A\right) = -\nabla_\parallel(\phi - \delta p_e) + \eta_\parallel \nabla_\perp^2 A - 4\mu_\perp^e \delta^2 \frac{m_e}{m_i}\nabla_\perp^4 A$$

$$\frac{d}{dt}n + \beta\frac{d}{dt}p = \beta[\Omega, \phi - \delta p_e] - \beta\nabla_\parallel(v + \delta\nabla_\perp^2 A) + \eta_\perp \beta \nabla_\perp^2 p$$

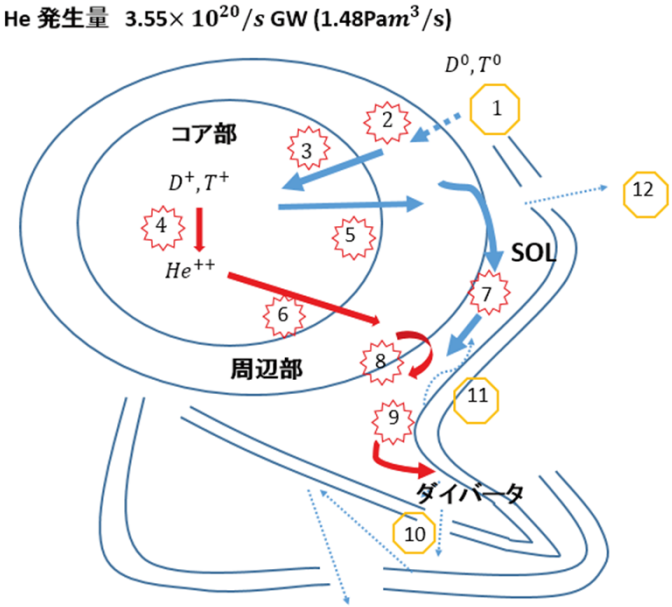
$$\frac{d}{dt}v = -\nabla_\parallel p + 4\mu_\perp^i \nabla_\perp^2 v$$

$$\frac{3}{2}\frac{dT_e}{dt} - \frac{\beta_e}{\beta}\frac{dn}{dt} = -\alpha_T \delta\beta_e \nabla_\parallel \nabla_\perp^2 A + \varepsilon^2 \chi_{e\parallel}\nabla_\parallel^2 T_e + \chi_{e\perp}\nabla_\perp^2 T_e$$

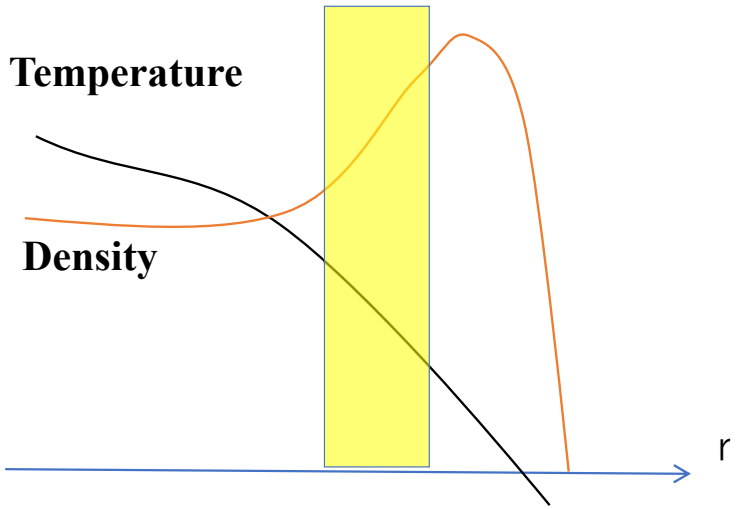
$$\frac{d}{dt} = \frac{\partial}{\partial t} + [\phi, ], \nabla_\parallel = \nabla_\parallel^{(0)} - [A, ], F = \phi + \delta p_i, \delta p_i = \delta_i n, \delta p_e = \delta T_e + \delta_e n, \delta_i = \frac{\beta_i}{\beta}\delta, \delta_e = \frac{\beta_e}{\beta}\delta$$

# Application of 5-field Reduced MHD Model

**Control of particle transport is an important issue for burn-control in ITER and DEMO.**




S. Matsuda, NIFS-MEMO-80 (2017)




- **Hollow density profile is often seen after pellet injection or gas-puff.**

**⇒inversed density gradient appears in the edge region.**

**Particle pinch mechanism for inverted density gradient in semi-collisional regime [Miki APTWG ‘ 17]**

**ITG/TEM stability is investigated for inverted density gradient in collisionless limit [Du ‘17]**

# Parallel Numerical Algorithms (1D Domain Decomposition)

$$\frac{\partial U}{\partial t} = \mathcal{L}(U) + \mathcal{N}(U,U)$$

**Linear terms are solved by Crank Nicolson implicit method and nonlinear terms are solved by Predictor-Corrector scheme.**

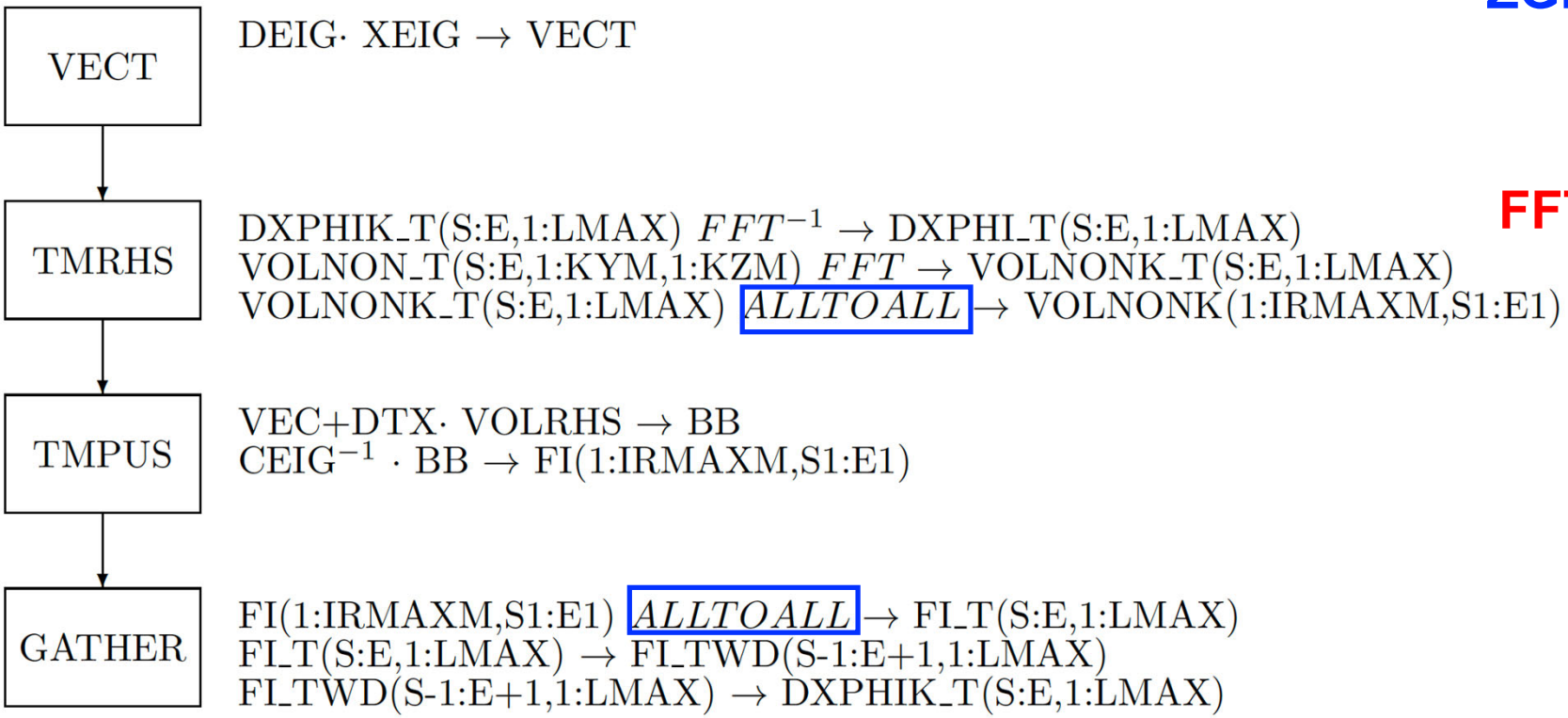
**Lapack**

**ZGBSV**

**Solve** $CEIG \cdot XEIG^{n+1} = DEIG \cdot XEIG^{n} + \Delta t \cdot RHS^{n}$

**Pseudo-spectral method**

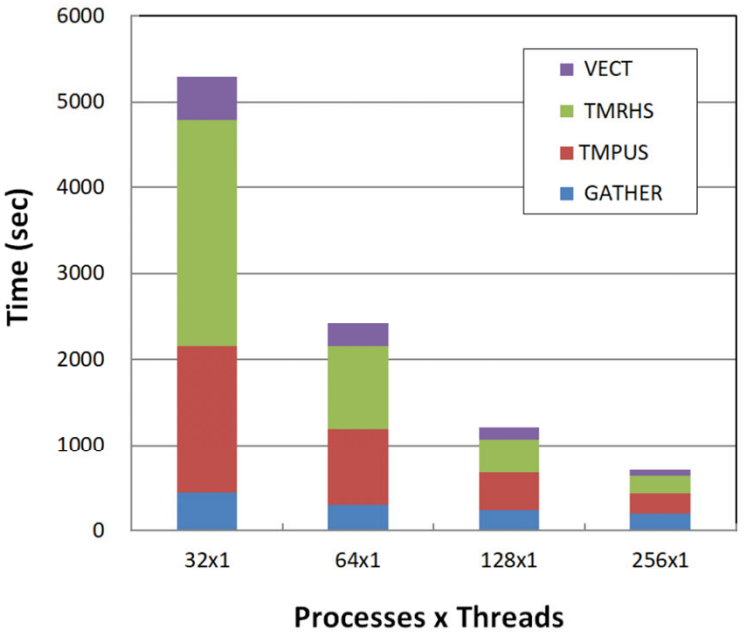
**FFTW**



# Cost Evaluation on HPE SGI8600 (NVIDIA V100)

```
main
|
|--call STMAIN
|--call ARRAY_SIZE, ARRAY_ALLOCG
|--call STINT
|--call ARRAY_ALLOC
|--call GATHER
|     |---call TRNS1_NEW
|     |---MPI_SENDRECV
|--call calc_eta(0)
|--call TMMAIN
          |--call TMENR(0)
          |-- do i=1,100,000,000
                |  !--FIRST STEP
                |-- call calc_eta(1)
                |-- call MATRIX
                |-- call VECT
                |-- call TMRHS
                |           |--- call TRNS2_NEW
                |           |--- CALL MPI_ALLGATHER
                |-- call TMPUS
                |-- call GATHER
                |   !--- SECOND STEP
                |-- call VECT
                |-- call TMRHS
                |-- call TMPUS
                |-- call GATHER
                |-- call CLEARM
```

**Main Loop**

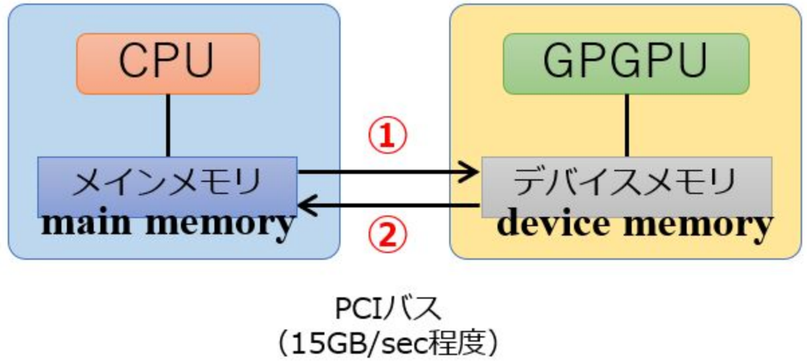


# Case with simply replacing FFTW with cuFFT

**1 node 40 process 1 thread 1GPU**

**FFTW**

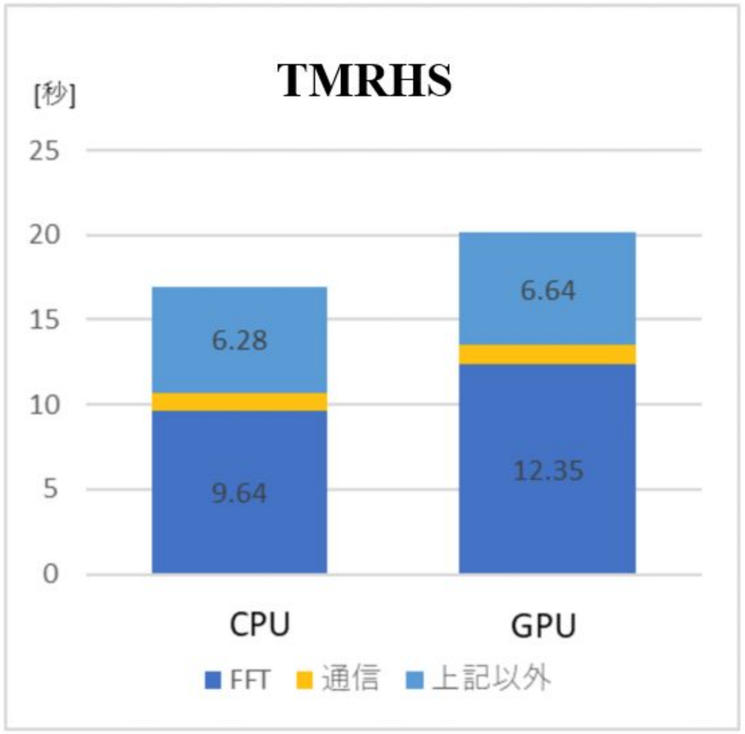
```
call dfftw_plan_dft_c2r_2d(plan(1),NY,NZ,...)
call dfftw_plan_dft_c2r_2d(plan(2),NY,NZ,...)
   :
DO I=S,E
  DO L=1,N
    call dfftw_execute(plan(L))
  END DO
END DO
```




PCIバス
（15GB/sec程度）

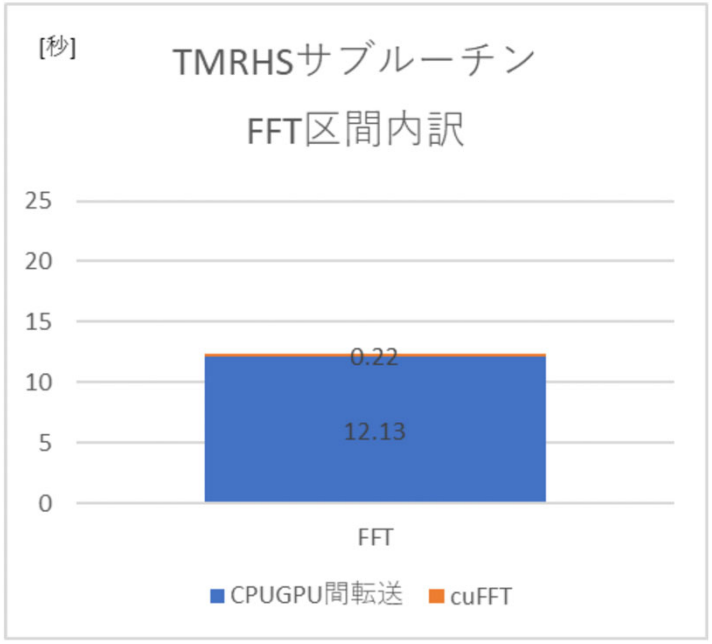
**cuFFT**

```
istat = cufftPlan2d(plan(1),NZ,NY,...)
istat = cufftPlan2d(plan(2),NZ,NY,...)
   :
DO I=S,N
  W1C_d=W1C
  W2C_d=W2C        ①
   :
  istat = cufftExecZ2D(plan(1),W1C_d,W1R_d)
  istat = cufftExecZ2D(plan(2),W2C_d,W2R_d)
   :
  W1R=W1R_d
  W2R=W2R_d        ②
END DO
```



```
(中略)
      if (myid.eq.0) t0=MPI_WTIME() ! okada
#ifdef CUFFT
       W1C_d =  W1C
(中略)
      if (myid.eq.0) then
          t1=MPI_WTIME() !hpe
          tgcom=tgcom + t1-t0 ! hpe
      endif
      istat = cufftExecZ2D(cuplan(1), W1C_d, W1R_d)
 (中略)
      if (myid.eq.0) then
          t2=MPI_WTIME() !hpe
          tfft=tfft + t2 - t1 ! hpe
      endif
       W1R = W1R_d
(中略)
      if (myid.eq.0) tgcom=tgcom + MPI_WTIME()-t2 ! hpe
#else
(中略)
#endif
      if (myid.eq.0) tife=tife + MPI_WTIME() - t0 ! okada
```

② ③ ② ①




**It is found that elapse time for data transfer between CPU and GPU is dominant in FFT area.**

**FFT library of GPU is faster than that of CPU: cuFFT 0.22sec vs FFTW 9.64 sec.**

```
main
|
|--call STMAIN
|--call ARRAY_SIZE, ARRAY_ALLOCG
|--call STINT
|--call ARRAY_ALLOC
|--call EQUIL
|--call GATHER
|    |---call TRNS1_NEW
|    |---MPI_SENDRECV
|--call calc_eta(0)
|--call TMMAIN
|--call TMENR(0)
|        |-- do i=1,100,000,000
|        |    !--FIRST STEP
|        |--- call calc_eta(1)
|        |--- call MATRIX
|        |--- call VECT
|        |--- call TMRHS
|        |        |--- call TRNS2_NEW
|        |        |--- CALL MPI_ALLGATHER
|        |--- call TMPUS
|        |--- call GATHER
|        |   !--- SECOND STEP
|        |--- call VECT
|        |--- call TMRHS
|        |--- call TMPUS
|        |--- call GATHER
|        |--- call CLEARM
```

① enter data, exit data directive

② update directive

③ kernels directive

!$acc enter data create(W1C(1:LYMED,1:KZMWD,S:E),…)

!$acc update device(DXVOLK_T(S:E,1:NDM),…)

!$acc host_data use_device(W1C(1:LYMWD,1:KZMWD,S:E),…)

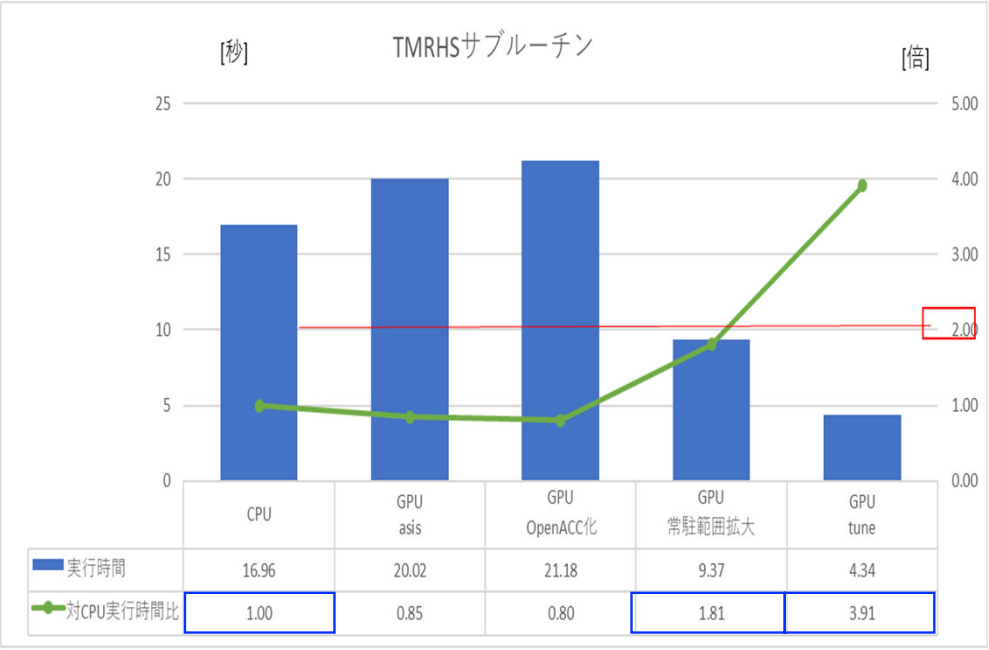
!$acc exit data delete(W1C,…)

```
!$acc data present( &
!$acc W1C(1:LYMWD,1:KZMWD,S:E), W2C(1:LYMWD,1:KZMWD,S:E),W3C(1:LYMWD,1:KZMWD,S:E),W4C(1:LYMWD,1:KZMWD,S:E), &
...
DO L=S1,E1
DO I=1,IRMAXM
VOLRHS(I,L)=0.0D0
...
END DO
END DO

IF(IST ==0) THEN
istat = cufftPlan2d(cuplan(1), KZMWD, KYMWD, CUFFT_Z2D)
...
IST=1
if (myid.eq.0) write(*,*) 'time fft_plan:', MPI_WTIME() - t0  ! okada
END IF
!$acc kernels
!$acc loop collapse(3)
DO I=S,E
 DO N=1,KZMWD
 DO M=1,LYMWD
W1C(M,N,I)   = (0.0D0,0.0D0)
...
 END DO
 END DO
END DO
!$acc loop independent private(IY,IZ) collapse(2)
DO I=S,E
 DO L=1,NDM
  IY=IPY(L)
  IZ=IPZ(L)
  W1C(IY,IZ)  = DXVOLK_T(I,L)
...
 END DO
END DO
!$acc end kernels
```

**Execution on GPU**

**Modified subroutine TMRHS**

```
!$acc host_data
use_device(W1C(1:LYMWD,1:KZMWD,S:E),W2C(1:LYMWD,1:KZMWD,S:E),W3C(1:LYMWD,1:KZMWD,S:E),W4C(1:LYMWD,1:KZMWD,S:E),&
…
DO I=S,E
  istat = cufftExecZ2D(cuplan(1), W1C(1,1,I), W1R(1,1,I))
…
END DO
!$acc end host_data
!$acc kernels
!$acc loop
DO I=S,E
  DO N=1,KZMWD
   DO M=1,KYMWD
     DXVOL_T(M,N,I) = W1R(M,N)
…

    END DO
  END DO
END DO
!$acc loop collapse(3)
DO I=S,E
  DO N=1, KZMWD
    DO M=1,KYMWD
VOLNON_T(M,N,I) = &
-DXPHI_T(M,N,I)*DYVOL_T(M,N,I)+DYPHI_T(M,N,I)*DXVOL_T(M,N,I)  &
+DXPSI_T(M,N,I)*DYCUR_T(M,N,I)-DYPSI_T(M,N,I)*DXCUR_T(M,N,I)
…
   END DO
  END DO
END DO
…
!$acc end kernels
!$acc update host(DENSNON_T,TEMENON_T)
call CALC_RADLOSS(30)
call CALC_SOURCE(IPTURB)
…
```




**Elapse time has been reduced from 21.18 sec to 9.37 sec for 1 GPU case (~1.81 times faster than CPU case)**

**Elapse time has been reduced from 9.37 sec to 4.34 sec for 4 GPU case (~3.91 times faster than CPU case)**

# Benchmark on ATOS BULLSEQUANA X2135 (NVIDIA A100)

# LEONARDO Booster, CINECA

| Model | Atos BullSequana X2135 "Da Vinci" single-node GPU blade |
|---|---|
| Racks | 116 |
| Nodes | 3456 |
| Processors | single socket 32 cores Intel Ice Lake CPU<br>1 x Intel Xeon Platinum 8358, 2.60GHz TDP 250W |
| Accelerators | 4 x NVIDIA Ampere GPUs/node, 64GB HBM2e NVLink 3.0 (200GB/s) |
| Cores | 32 cores/node |
| RAM | 512 (8x64) GB DDR4 3200 MHz |
| Peak Performance | about 309 Pflop/s |
| Internal Network | DragonFly+ 200 Gbps (NVIDIA Mellanox Infiniband HDR) 2 x dual port HDR100 per node |
| Storage (raw capacity) | 137.6 PB based on DDN ES7990X and Hard Drive Disks (Capacity Tier)<br>5.7 PB based on DDN ES400NVX2 and Solid State Drives (Fast Tier) |

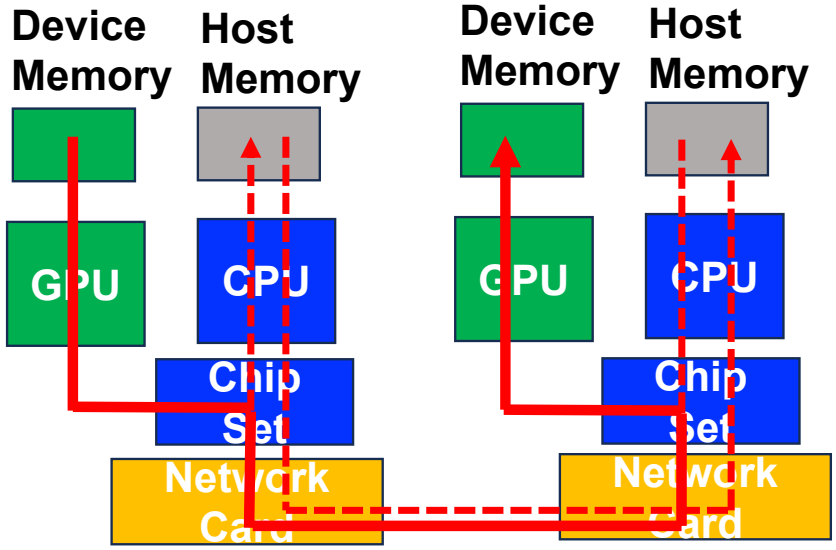
| | CPU 32MPI | 4MPIx4GPU (1 node) | 8MPIx4GPU (1 node) |
|---|---|---|---|
| matx | 0.37 (sec) | 1.54 | 0.88 |
| vect | 5.62 | 23.86 | 12.96 |
| rhs | 11.90 | 6.58 | 5.08 |
| fft | 4.24 | 1.88 | 0.61 |
| com | 0.95 | 1.31 | 0.76 |
| pus | 27.49 | 164.95 | 83.13 |
| gthr | 4.90 | 10.59 | 6.23 |
| com | 2.90 | 6.24 | 2.55 |
| totoal | 93.17 | 241.11 | 146.96 |

**The sparse matrix inversion part (LAPACK) is expensive.**

# GPU Communications between Nodes

In TMRHS (rhs) subroutine, MPI communication is performed using NVLINK+RDMA.
For comparison to communication via host memory, 2 nodes are used for benchmark.

| | via Host Memory | NVLINK+RDMA |
|---|---|---|
| rhs | 8.70 (sec) | 4.49 |
| fft | 0.31 | 0.31 |
| com | 4.07 | 2.27 |




```
!$acc host_data use_device(VOLNONK_T,VOLNONK)
CALL MPI_ALLTOALLW(VOLNONK_T, IRCNT, IRDISP, IRTYPE, &
                   VOLNONK,   ISCNT, ISDISP, ISTYPE, COMM1D, IERR)
```

# Tunning by NVIDIA

## Optimization I

### Diagnosis

Relatively small size of 2D FFT is executed many times which implies low level of GPU utilization rate

### Countermeasure

P xoldsh#5G#IWv#luh#{hfxwh#rq#JSX#

## Optimization II

### Diagnosis

Pre-process and post-process of Alltoall communication are not parallelized

### Countermeasure

Parallelization by OpenACC and execution on GPUs

## Optimization III

### Diagnosis

Data copy between CPUs and GPUs before and after Alltoall communication

### Countermeasure

Direct Alltoall communication by NCCL

**Test Environment**

DGX H100, 4GPUs

**Execution conditions**

4MPI x 4GPU

**Asis**: cuFFT on GPU, Alltoall communication, pre-process and post-process on CPU

**Batched FFT**: optimization I and optimization II

**NCCL**: optimization I, optimization II and optimization III

| | Asis | Batched FFT | NCCL |
|---|---|---|---|
| **Time: rhs** | **24.05** | **10.69** | **3.44** |

# Summary and Discussion

**The optimization of R5F code for GPU version is performed using OpenACC.**

- ✓ **TMRHS (rhs) subroutine is tuned which is the most expensive part of the code.**
- ✓ **We have reduced communications between CPU and GPU in between cuFFT calls.**
- ✓ **In addition, we have extended 4 GPU use in 1 node.**

**As the result, elapse time of subroutine TMRHS is reduced more than half.**

**Supercomputer LEONARDO in CINECA is also used for benchmark of R5F.**

- ✓ **Sparse matrix inversion part (pus) by LAPACK is most expensive part, it should be ported in GPU side.**
- ✓ **It is shown that NVLINK gives better performance for MPI communication compared with IB.**

**In future work, we should implement**

**fftw_plan_many_dft => cufftPlanMany** **ZGBSV => cuSPARSE**